AS-MCOE: Tutor Inteligente para o Processo Ensino-Aprendizagem

Alexandre de O. Zamberlam[1], Rodrigo R. V. Goulart[2], Cláudia C. C. Perez[3], Lucas E. Freitas[4], Janaina M. Blanco[5], Marcus Hübner[6], Selenir C. G. Kronbauer[7], Lúcia M. M. Giraffa[8]

Resumo

O artigo relata a pesquisa realizada no projeto AS-MCOE. A metodologia e os resultados do experimento com alunos do ensino fundamental são apresentados, bem como a tecnologia utilizada para a construção do jogo e o processo de conversão de agentes BDI modelados/implementados em X-BDI para a linguagem AgentSpeak(L) interpretada no Jason.

Palavras-chave: IA. Agentes Cognitivos. Agentes BDI. AgentSpeak(L). Jason, STI.

Abstract

This paper describes the investigation done in AS-MCOE project. The methodology and the experiment results with students of the preliminary-instruction are presented, as well as the technology used to build the game and the set of efforts done to rewrite the project AS-MCOE from X-BDI approach to AgentSpeak language in Jason.

Keywords: AI. Cognitive Agents. BDI Agents. AgentSpeak(L). Jason. ITS.

[1]Membro do Grupo de Pesquisa em Computação Aplicada da Feevale. E-mail: alexz@feevale.br.
[2]Membro do Grupo de Pesquisa em Computação Aplicada da Feevale. E-mail: rodrigo@feevale.br.
[3]Membro do Grupo de Pesquisa em Computação Aplicada da Feevale e da Escola de Aplicação Feevale. E-mail: claudiaperez@feevale.br.
[4]Membro do Grupo de Pesquisa em Computação Aplicada da Feevale. E-mail: knight86@feevale.br.
[5]Membro da Escola de Aplicação Feevale. E-mail: janainamb@feevale.br.
[6]Membro da Escola de Aplicação Feevale. E-mail: 0000837@feevale.br.
[7]Membro da Escola de Aplicação Feevale. E-mail: selenir@feevale.br.
[8]Membro da PUCRS Virtual. E-mail: giraffa@pucrs.br.

Introdução

O projeto AS-MCOE (AgentSpeak MCOE) tem como objeto pesquisar os jogos educacionais, que se destacam pela popularidade e inserção na comunidade infanto-juvenil e que até hoje não agregam a assistência artificial como um diferencial comercial e/ou pedagógico. Essa investigação, em curso, busca desenvolver um estudo sobre a utilização da abordagem BDI (*Belief, Desire, Intention*) em um jogo educacional, a fim de avaliar o processo de desenvolvimento de habilidades cognitivas e sociais em alunos do Ensino Fundamental.

O tema do jogo, proposto inicialmente em Giraffa (1999), é sobre Ecologia e baseia-se na interação do aluno (via um personagem) em uma cadeia alimentar de um lago, composta também por personagens artificiais inteligentes. O objetivo do jogo é manter o equilíbrio da cadeia por meio de "poderes" ou ferramentas que o personagem/aluno possui, sendo que as dificuldades relacionadas ao jogo e/ou ao conteúdo são avaliadas por um Tutor Artificial (especificado através da abordagem BDI), que envia mensagens com dicas e explicações ao aluno.

Neste trabalho, a modelagem utilizada por Giraffa (1999) e proposta em Móra (2000), conhecida como X-BDI, foi substituída pela de Rao (1996), cuja linguagem, AgentSpeak(L), foi implementada por Hübner (2004). Esses novos recursos viabilizaram a reimplementação dos experimentos realizados por Giraffa e a sua posterior avaliação. Além disso, novos recursos foram utilizados para o desenvolvimento do jogo: a linguagem Java e o *engine* GTGE - *Golden T Game Engine* (Studios, 2006).

Este artigo descreve o projeto AS-MCOE, o processo de conversão da especificação dos agentes do jogo (de X-BDI para AgentSpeak(L), executado no interpretador Jason e os resultados alcançados com um experimento realizado com alunos da 3ª série do ensino fundamental.

A fim de facilitar a compreensão do processo e permitir sua reprodução por outros grupos de pesquisa, o texto está dividido em 10 seções. A seção 2 detalha o histórico da pesquisa realizada nos últimos onze anos. A seção 3 aborda Sistemas Tutores Inteligentes e o papel de agentes inteligentes nesse domínio. A seção 4 discute acerca da arquitetura BDI, suas possibilidades e características. As seções 5 e 6 apresentam, respectivamente, o ambiente X-BDI e a linguagem AgentSpeak(L), bem como seu interpretador Jason e o engine GTGE. A seção 7 detalha o processo de conversão. Na seção 8, a arquitetura do jogo é descrita. A seção 9 trata sobre o experimento realizado, seguida pelas considerações finais. No final do texto, encontram-se as referências utilizadas para a elaboração deste artigo.

1. Histórico

AS-MCOE faz parte de um conjunto de pesquisas sobre *softwares* educacionais inteligentes realizadas nos últimos dez anos. O eixo principal de todas elas é o *Multi Cooperative Environment* - MCOE, proposto em Giraffa (1999), que descreve um ambiente composto por um jogo de ecologia e um tutor inteligente.

Essa investigação teve início com o jogo Eco-lógico (1996), que foi um trabalho de conclusão de curso de graduação (Raabe, 1996), construído com a ferramenta ToolBook (ASYMETRIX, 1994), e que trabalhava somente com a apresentação de uma cadeia alimentar, exibindo conceitos e algumas relações. Em 1998, iniciou-se o projeto MCOE, no qual o jogo apresenta ao aluno um ambiente em que aparecem inúmeros problemas. Ao longo de sua interação com o programa, o aluno deve solucioná-los por meio de seu conhecimento prévio e pelo uso de ferramentas que, combinadas, auxiliarão na

construção de uma estratégia de ação. O jogo é composto por um lago onde existe um ecossistema equilibrado formado por peixes, plantas e microrganismos, até que a intervenção de poluentes provoquem alterações no seu estado normal. Esses poluentes aparecem de forma aleatória ao longo do jogo e são combatidos através de ferramentas do personagem, escolhidas livremente por cada aluno para resolver o problema da poluição do lago. O jogo foi concebido para alunos que estejam freqüentando a 3ª e 4ª séries do ensino fundamental e foi construído em Visual C++ e com o pacote de APIs DirectX (Giraffa, 1999). Já o tutor – construído obedecendo ao paradigma de agentes BDI – especificado e implementado por meio do ambiente X-BDI (Mora, 2000), avalia o relacionamento do jogador com o jogo, auxiliando o aluno com mensagens de ajuda. O trabalho desenvolvido por Giraffa (1999) reservou-se a explorar o ensino da conscientização ecológica com o auxílio de um "tutor" artificial através de um ambiente que utiliza a metáfora de jogos. O problema abordado e as soluções apresentadas trouxeram desafios a serem trabalhados.

Na fase seguinte, Callegari (1999) apresentou uma proposta (RL-MCOE) de ampliação da arquitetura multiagente reativa existente no ambiente MCOE, por meio de uma técnica de aprendizagem por reforço (*W-Learning*).

Em 2001, foi apresentado o E-MCOE (Goulart, 2002), que introduziu um agente mediador no MCOE, acompanhando as trocas de informações entre o jogo e o tutor. Essa funcionalidade também utilizou agentes reativos modelados com a técnica de aprendizagem por reforço proposta por Callegari (1999). No entanto, os agentes aluno e tutor foram codificados de acordo com o modelo BDI de agentes cognitivos. Nesse trabalho, foram levados em consideração tarefas/processos externos que normalmente eram tratados no módulo tutor. Esses processos permitem distribuir a complexidade da modelagem e supervisão do conjunto de informações entre a interação dos agentes.

Por fim, em 2005, iniciou-se o projeto AS-MCOE (Figura 1), que se caracteriza pela investigação científica dos avanços pedagógicos e tecnológicos decorrentes da utilização de técnicas de Inteligência Artificial na educação de crianças, tendo como objetivo avaliar a utilização de tutores inteligentes no ensino de ecologia, dando continuidade ao trabalho desenvolvido no ambiente MCOE. Entretanto, o tutor e os agentes existentes estão sendo implementados por meio da linguagem AgentSpeak(L) e do interpretador Jason. Esses agentes estão inseridos em um jogo, construído através da *Golden T Game Engine* (GTGE).

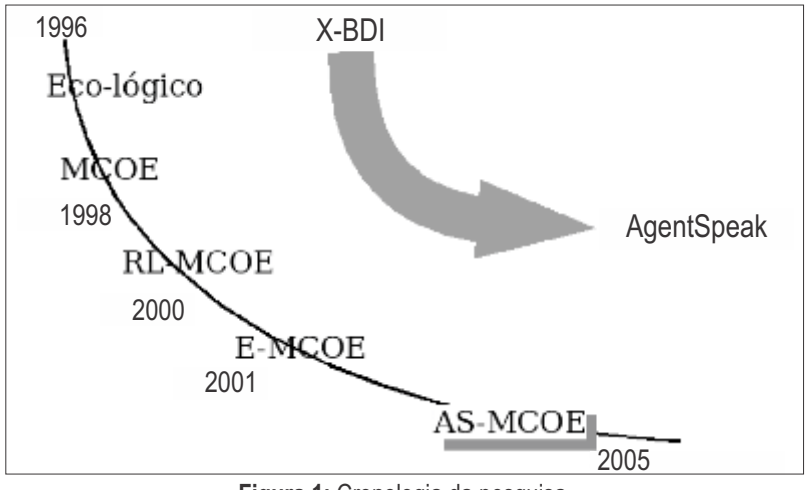

Figura 1: Cronologia da pesquisa.

2. Sistemas Tutores Inteligentes

Sistemas Tutores Inteligentes (STI) são *softwares* educacionais que se diferem das demais aplicações voltadas ao ensino, pela inclusão de uso de técnicas oriundas da Inteligência Artificial, que especificam o que ensinar, a quem ensinar e de que forma ensinar, expressas numa arquitetura modular, onde se busca modelar o estado cognitivo corrente do aprendiz, a fim de possibilitar atendimento personalizado ao aluno/usuário do sistema (Giraffa, 1999; Murray, 1999). Há algum tempo, conceitos de agentes inteligentes têm sido aplicados a esses sistemas, como forma de ampliar suas possibilidades face ao advento da Internet e dos sistemas distribuídos.

A inclusão da abordagem de agentes permite fazer uma releitura na arquitetura tradicional de STI, onde os módulos podem ser substituídos por sociedades de agentes que trabalham de forma coordenada para atender o aluno. Nesse contexto, a modelagem do aluno pode ser realizada de diversas formas, sendo uma delas através da utilização de agentes cognitivos. Segundo Wooldridge (2000), um agente inteligente pode ser definido como uma entidade de *software* de comportamento autônomo, que está inserido em algum ambiente sobre o qual é capaz de realizar ações para alcançar objetivos de projeto e perceber as alterações. Portanto, *softwares* educacionais podem utilizar a tecnologia de agentes, pois possuem essas características.

A possibilidade de utilização de agentes inteligentes em STI traz vantagem em relação aos tradicionais STI, por apresentarem uma flexibilidade maior no tratamento dos elementos que compõem o sistema. De acordo com Bolzan (2002), com a inserção da tecnologia de agentes no projeto e desenvolvimento de STI, verifica-se que o conteúdo do domínio passa a ser variado e não apenas lógico, bem estruturado e restrito (ensino de Matemática, Física, e outros). O conteúdo passa a ser modelado independentemente do domínio. Para maiores detalhes sobre STI com agentes pedagógicos baseados em BDI, sugere-se a leitura de Giraffa (1999).

3. Arquitetura BDI

Essa arquitetura de agentes cognitivos pode ser estruturada, segundo Bordini e Vieira (2003), através das crenças que representam aquilo que o agente sabe sobre o estado do ambiente e dos agentes que atuam naquele ambiente (inclusive sobre si mesmo). Os desejos representam estados do mundo que o agente quer atingir (são representações daquilo que ele quer que passe a ser verdadeiro no ambiente). Assim, desejos podem ser contraditórios, ou seja, é possível desejar coisas que são mutuamente exclusivas do ponto de vista de ação prática. Normalmente, os objetivos são referenciados como um subconjunto dos desejos, que são todos compatíveis entre si. As intenções representam seqüências de ações específicas que um agente se compromete em executar para atingir determinados objetivos (Bratman, 1990). Algumas abordagens de arquitetura BDI agregam noções de planos e objetivos, como, por exemplo, os trabalhos de Bratman (1990), Rao (1996) e Cohen e Levesque (1987).

A atualização dos objetivos se dá de duas formas: I) as observações do ambiente possivelmente determinam novos objetivos do agente; II) a execução de intenções de mais alto nível pode gerar a necessidade de que objetivos mais específicos sejam atingidos.

Assim sendo, o princípio dessa abordagem é descrever o processamento interno de um agente por meio de um conjunto de estados mentais, como crença, desejo e intenção, e definir uma estrutura de controle através da qual o agente seleciona racionalmente o curso de suas ações. De fato, o que

caracteriza um agente, tendo por base uma visão mais ampla, são as interações que ele realiza com o ambiente e os processos internos que possibilitam tais interações (Giraffa, 1999).

Dentro da abordagem BDI de agentes, há poucos ambientes de programação que realmente fornecem suporte adequado à programação de sistemas baseados nessa categoria de agentes. Entre as ferramentas disponibilizadas para a especificação e a programação de agentes BDI, pode-se citar o X-BDI (*eXecutable* BDI) (Móra, 2000) e o interpretador Jason (Hübner et al., 2004), que implementa a linguagem AgentSpeak(L) (Rao, 1996). O ambiente X-BDI foi utilizado para modelar o tutor do ambiente MCOE (*Multi Cooperative Environment*) (Giraffa, 1999). E a linguagem AgentSpeak(L) e o seu interpretador Jason (no projeto AS-MCOE) para remodelagem do tutor nessa proposta, que atualmente tem-se mostrado mais promissora.

4. *eXecutable* BDI

X-BDI é um ambiente que permite a descrição formal de agentes cognitivos baseados na abordagem BDI, sendo ao mesmo tempo uma linguagem para implementação desses agentes (Móra, 2000).

Derivada do modelo formal de agentes BDI, o X-BDI implementa os algoritmos que tratam das interações desses estados mentais: como manter a consistência da base de crenças; como manter as intenções consistentes internamente com as crenças; como derivar intenções de desejos e planos de intenções. Portanto, X-BDI é uma ferramenta que, "alimentada" com a descrição de um agente, em termos de suas crenças e desejos, é capaz de gerenciar esses estados mentais e produzir seqüências de ações que satisfaçam esses desejos do agente inserido em um ambiente (Zamberlam e Giraffa, 2002).

O X-BDI não é visto como um agente completo, mas como o *kernel* cognitivo, que é parte de um agente. Implementado em Prolog, o *kernel* X-BDI é um processo que interage com a parte de sensoriamento e de atuação de um agente. Cada informação que deve ser percebida pelo agente, como uma crença, é transmitida para esse núcleo. Esse, por sua vez, conserva as mudanças no ambiente e informa os planos adequados. Assim, o X-BDI é um ambiente de modelagem e uma linguagem de programação de agentes BDI.

Crenças são representadas pelo predicado *bel:* ***bel(agente, propriedade, tempo)***. A especificação do *tempo* é opcional. A possibilidade de incluir tempo no formalismo do X-BDI permite representar crenças no futuro (expectativas do agente).

Ações, por sua vez, são representadas pelo predicado *act:* ***act(agente, ação)***. A identificação do agente é opcional, podendo ser representadas expressões do tipo:

$$act\ (agente,\ ação)\ causes\ propriedade_1,...,\ propriedade_n$$
$$if\ propriedade_1,\ ...,\ propriedade_n$$

Portanto, ações têm: nome, argumento (atributo), pós-condições (efeitos) e pré-condições (requisitos).

Desejos utilizam-se do predicado *des:* ***des(agente, propriedade, prioridade)***. A especificação de uma prioridade é facultativa e, se especificada, deve representar um valor entre um e zero (inclusive).

5. AgentSpeak(L), Jason e GTGE

A linguagem AgentSpeak(L) possibilita a especificação de crenças, objetivos e planos para agentes BDI. AgentSpeak(L), conforme Rao (1996), é uma extensão de programação em lógica, definida para arquitetura de agentes cognitivos baseados em crenças, desejos e intenções. Enfim, é uma linguagem que é utilizada na construção de agentes BDI em sistemas que estão permanentemente em execução (*reactive planning systems*), reagindo a eventos que ocorrem no ambiente (Bordini e Vieira, 2003).

Um agente AgentSpeak corresponde à especificação de um conjunto de crenças e um conjunto de planos. As informações sobre os desejos (estados futuros a serem atingidos), além das alternativas disponíveis ao agente para ativar as intenções (atingir seus objetivos), estão implicitamente representadas nos planos (Hübner, 2004). Pode-se afirmar, então, que planos são referências às ações básicas de agentes no ambiente, ou seja, um plano determina uma forma de atingir um determinado objetivo (cabeça do plano).

Um plano, conforme Hübner *et al.* (2004), é formado por um evento ativador interno/externo (denotando o propósito do plano); seguido pela conjunção de crenças (literais), representando o contexto (conseqüência lógica das crenças correntes do agente para um plano aplicável, ou pré-condição); o restante do plano é composto por ações básicas, que o agente deve executar e/ou subplanos (realizações ou testes de verificação), isto é, objetivos intermediários que o agente deve tentar atingir, para que o plano tenha sucesso. A satisfação de uma pré-condição e a ocorrência de um evento ativador são as condições para que um plano possa entrar em execução.

Além disso, para a implementação de um agente AgentSpeak é necessária a especificação das funções de seleção de eventos, planos aplicáveis e as intenções (Hübner *et al*., 2004). A Figura 2 apresenta a gramática da linguagem.

```
agente     ::= crenças  planos
crenças    ::= crença1 . . . crençaK
planos     ::= plano1 . . . planoN
plano      ::= evento : contexto -> corpo
evento     ::= + crença | ? crença | +! objetivo
contexto   ::= crença | not(crença) | contexto & contexto | true
corpo      ::= objetivo | ?objetivo | +crença | -crença | ação | corpo; corpo
crença     ::= atomo
objetivo   ::= atomo
atomo      ::= Predicado(termo1 , . . . termoM)
```

Figura 2: Gramática de AgentSpeak(L), com K, N $\geq$ 1, M $\geq$ 0 (Bordini e Vieira, 2003).

5.1 O interpretador Jason

O interpretador Jason (Bordini e Hübner, 2004) é um sistema para modelagem e execução de agentes AgentSpeak. Nele é possível descrever planos, crenças e o ambiente em que os agentes estão inseridos. É distribuído na forma de um *plugin* da Ferramenta de Desenvolvimento Integrado (IDE) Jedit, que viabiliza mecanismos para facilitar a configuração, codificação e execução de um sistema multiagente.

De acordo com Hübner *et al*. (2004), o interpretador implementa atos de fala (*speech act*) para a comunicação entre agentes por meio do ambiente SACI (*Simple Agent Communication Infrastructure*), tornando possível a utilização de um sistema multiagente em Jason em redes de computadores. Ele também possui uma biblioteca de ações internas, que podem ser referidas nos planos dos agentes. Entre elas, destacam-se:

.send -	usada para enviar uma mensagem a um agente;
.broadcast -	envia mensagens a todos agentes da sociedade;
.print -	imprime mensagem no console.

Outra importante característica em relação a outros interpretadores de agentes BDI é que ele foi implementado na linguagem Java, estando disponível em código aberto (*Open Source*) e distribuído sob a licença GNU LGPL.

5.2 *Golden T Game Engine*

O *engine* GTGE, conforme Studios (2006), é uma biblioteca/API de programação multiplataforma para o desenvolvimento de jogos. Escrita na linguagem Java, ela fornece um conjunto de rotinas de alto nível para programação orientada a jogos, a fim de que o programador tenha fácil acesso a recursos de *hardware* (como placas aceleradoras de vídeo e som) e rotinas básicas de jogos (como detecção de colisões e de comportamento de personagens).

Afinal, as três últimas ferramentas descritas pertencem à categoria de *software* livre, que, segundo a definição criada pela *Free Software Foundation*, é qualquer programa de computador que pode ser usado, copiado, estudado, modificado e redistribuído com algumas restrições.

6. O processo de conversão

O tutor inteligente proposto em Giraffa (1999) foi traduzido para a linguagem AgentSpeak(L), em que questões particulares de ambas as linguagens tiveram de ser avaliadas. Na linguagem X-BDI, onde o tutor foi originalmente modelado, os desejos são codificados de forma explícita e os planos que permitem realizar esses desejos são formados pelo conjunto de ações e/ou crenças que se tornam realidade mediante um conjunto de condições (contexto).

Por exemplo, o tutor deseja ajudar os alunos, ou seja, almeja o estado de que o aluno esteja auxiliado – *des(tutor, alunoAuxiliado(ALUNO)).*

A maneira de concretizar esse desejo é gerar uma ação que envia uma mensagem ou mensagens de aviso para o aluno, quando necessário, causando as crenças de que o agente realizou a ajuda e que enviou determinada mensagem. Os conteúdos dessas mensagens dependerão da estratégia adotada no momento pelo tutor. Mais detalhes sobre os conteúdos das mensagens em Giraffa (1999). A Figura 3 apresenta o código X-BDI.

```
des(tutor, alunoAuxiliado(ALUNO)) if
   bel(tutor, baixaInteratividade(ALUNO)).
act(tutor, enviarMensagem(X, ALUNO)) causes
   bel(tutor, mensagemEnviada(ALUNO)),
   bel(tutor, alunoAuxiliado(ALUNO).
bel(tutor, mensagemEnviada(ALUNO)) if
   bel(tutor, next(bel(ALUNO, peçoAjuda))),
   bel (tutor, energiaEcometro(Ee)),
   Ee >= 70,
   mensagem(X,ALUNO),
       X = 16.
```

Figura 3: Código X-BDI atualizado.

Porém a crença de que o tutor enviou uma mensagem – *bel(tutor, mensagemEnviada(ALUNO))* torna-se verdadeira somente quando as seguintes condições são satisfeitas:

1. o aluno acredita que vai receber ajuda – *bel(aluno, peçoAjuda)*;

2. o nível do ecômetro (Ee) é bom e está acima ou igual a 70;

3. a ação de enviar mensagem número 16 foi realizada. Observa-se que a crença de que o aluno acredita que irá receber ajuda é uma condição futura, ou seja, uma pré-condição ou contexto daquela crença.

Para modelar os desejos e as crenças do tutor e dos demais agentes em AgentSpeak(L), deve-se considerar que não há uma representação explícita de desejos nessa linguagem. Desejos são, em AgentSpeak(L), objetivos que podem ser realizados por planos ou subplanos, isto é, planos ativados por eventos internos.

```
+not(alunoAuxiliado(aluno1)).
+baixaInteratividade[source(ALUNO)]: not(alunoAuxiliado(ALUNO))
   <- !enviarMensagem(ALUNO),
      +alunoAuxiliado(ALUNO).
+enviarMensagem(ALUNO) : energiaEcometro(Ee) & Ee >= 70,
                                peçoAjuda(ALUNO)
   <- mensagem(16).
```

Figura 4: Código AgentSpeak(L).

A sintaxe do código da Figura 4 descreve um plano (*baixaInterativiidade[source(ALUNO)]*) para o evento externo em que o aluno – *source(ALUNO)* – que também é considerado um agente, necessita de ajuda do tutor, pois não está interagindo, por alguma razão, com o jogo. O tutor, então, verifica as pré-condições para auxiliar o aluno e chama um sub-plano para enviar uma mensagem específica para aquele aluno (*!enviarMensagem(ALUNO)*), e finalmente adiciona a crença (+*alunoAuxiliado(ALUNO)*), na base de crenças, de que aquele aluno foi ajudado. O subplano para enviar uma mensagem só será realizado, se o contexto for verdadeiro (*energiaEcometro(Ee) & Ee >= 70, peçoAjuda(ALUNO)*). Ao final, como ação básica do subplano, há o envio da mensagem 16.

Em resumo, a metodologia proposta para a conversão de X-BDI para AgentSpeak(L) é composta pelas seguintes etapas:

1. localizar o conjunto de ações, seus efeitos (pós-condições) e seus requisitos (pré-condições, ou contexto em AgentSpeak(L);

2. para cada ação ou conjunto de ações escritas em X-BDI, codificar em AgentSpeak(L) um plano que defina o curso dessas ações. De acordo com a gramática apresentada na Figura 2, a cabeça do plano é um evento ativador interno/externo, podendo ser uma adição ou remoção de crença ou a realização de um objetivo, isto é, um desejo selecionado à execução;

3. desejo é um estado motivacional que tende a produzir, modificar ou selecionar ações à luz das crenças. Então, é necessário codificar as crenças que ativam esses desejos (objetivos), ou seja, o contexto em AgentSpeak(L).

É importante ressaltar que um plano em AgentSpeak(L) pode ser, ao final de sua execução, uma crença (removida ou adicionada na base de crenças) ou um desejo atingido. Também deve ser levado em consideração que quando crenças são tratadas, é fundamental ter em mente que elas são a representação do conhecimento que o agente possui sobre o ambiente. Assim, em alguns casos, é necessária a codificação, em primeiro lugar, das crenças iniciais do agente, ou seja, qual o conhecimento inicial do agente em relação ao ambiente em que ele está inserido. E isso segue a mesma regra, tanto em X-BDI, quanto em AgentSpeak(L).

7. A Arquitetura do Jogo

O desenvolvimento de um jogo implica a especificação detalhada dos personagens, suas ações e percepções (ambiente), além das leis que regem o ambiente. Nesse caso, as ações e percepções são determinadas pelo papel de cada personagem. Um peixe pode, por exemplo, mover-se e perseguir outros seres que sejam considerados alimento. Um personagem poluidor (que não é "inteligente") e as ferramentas ou "poderes" do jogador influenciam na saúde da cadeia alimentar. Todos esses fatores determinam o grau de limpeza do lago, representado no jogo por um objeto chamado ecômetro (medidor), que é calculado a partir da média da energia dos seres vivos. Parte dessas heurísticas é mostrada no Quadro 1, de acordo com um estudo realizado por especialistas, relatado em Giraffa (1999).

	Retirada			Reposição	
Personagem vs. Interventor	**Lixo urbano**	**Pesca predatória**	**Esgoto doméstico**	**Pescar em área permitida**	**Devolver peixes em reprodução**
Peixe pequeno	-5	-30	-10	+10	+20
Peixe médio	-5	-20	-10	+10	+20
Peixe grande	-5	-10	-10	+10	+15
Plantas	-10	-	-10	+5	+5

Quadro 1: Retirada e reposição de energia.

O jogo é implementado seguindo as especificações de "jogabilidade", interface gráfica e ajuda *online* propostos originalmente no projeto MCOE. Apenas o *design* gráfico foi aprimorado com o auxílio de profissionais da área de comunicação que fazem parte da instituição de ensino onde a pesquisa é realizada. A Figura 5 ilustra o protótipo desenvolvido.

Para esse jogo, foram desenvolvidos mecanismos específicos para o tema em questão. O comportamento dos peixes foi modelado e implementado na forma de agentes cuja "área de percepção" influencia as ações a serem tomadas por eles. Por exemplo, um peixe de tamanho "Grande" persegue peixes de tamanho "Médio", quando ele está com fome e estes estão dentro de sua área de percepção. A avaliação da área de percepção de cada personagem é feita durante a atualização (*update*) de cada quadro da animação.

Figura 5: A interface do protótipo.

O processo de desenvolvimento do jogo seguiu três etapas:

1. análise e (re)modelagem da arquitetura e do jogo MCOE;
2. implementação do jogo e do tutor inteligente;
3. avaliação do jogo e do tutor no ambiente escolar.

A primeira etapa da pesquisa já foi realizada e concentrou-se no estudo de questões tecnológicas e operacionais. A segunda etapa foi subdividida na implementação do jogo e do tutor. Para avaliar o emprego do tutor na terceira etapa, foi desenvolvido um experimento, detalhado na próxima seção, contendo apenas o jogo.

8. Especificação de um Modelo de Experimento em Sala de Aula e Avaliação dos Resultados.

Com a finalidade de avaliar a influência do jogo sem o auxílio de um tutor artificial, um experimento em sala de aula foi planejado e executado no final do ano de 2006.

8.1 Estrutura do Experimento

O experimento foi realizado nos dias 18 e 21 de dezembro de 2006. A turma escolhida para o estudo foi do terceiro ano do ensino fundamental, composta por quinze alunos. No primeiro encontro, o questionário de pré-análise foi aplicado a todos os alunos. No segundo dia, os alunos foram divididos em dois grupos (A e B). O grupo A (sete alunos) era composto pelos alunos que tiveram pior desempenho na pré-avaliação. Eles brincaram com o jogo desenvolvido e em seguida responderam a um novo questionário. Os alunos do grupo B (oito alunos), que tiveram aproveitamento satisfatório, responderam ao novo questionário e então utilizaram o jogo. Esse experimento foi composto pelas seguintes etapas:

1. Definir o conteúdo a ser avaliado: com o auxílio dos professores de Biologia (e/ou disciplinas correspondentes ou relacionadas), da orientadora pedagógica da escola e das especificações descritas no trabalho de Giraffa (1999), foi definido o conteúdo de ecologia a ser abordado na avaliação. Como resultado, produziu-se uma lista de categorias de habilidades (Quadro 2) a serem verificadas. Para cada habilidade, foram associados alguns conceitos gerais e específicos.

2. Produzir questões de avaliação: novamente com auxílio dos professores e da orientadora, baseando-se no Quadro 2, foram criadas as questões para avaliação. Para tanto, foram elaboradas diferentes questões de mesma categoria, a fim de construir mais de um questionário.

3. Aplicar um questionário (pré-avaliação) na turma antes do uso do jogo: conforme indicado por Giraffa (1999) e em discussão com os professores de Biologia e a orientadora pedagógica, escolheu-se uma turma de alunos do terceiro ano do ensino fundamental. Aplicou-se, então, um questionário baseado na lista de categorias de habilidades e na lista de questões elaboradas anteriormente, com a intenção de verificar o conhecimento prévio dos alunos sobre ecologia. Foram utilizadas 10 questões.

	Habilidades desejadas	Conceitos Gerais	Conceitos Específicos
A	Relacionar os efeitos de um agente com um grupo de seres de uma cadeia	Agentes poluidores *versus* Seres vivos	Jet-ski, lanchas, peixes, microorganismos
B	Relacionar os conhecimentos sobre um recurso natural	Natureza, seres vivos	Peixes, aguapés, plantas, ecossistema
C	Relacionar os efeitos de um agente com um recurso natural	Agentes poluidores *versus* Natureza	Esgoto, lago, resíduos, ecossistema
D	Relacionar mecanismos ou atores de combate com os agentes poluidores a serem combatidos	Agentes poluidores *versus* Personagens jogo; Agentes poluidores *versus* Natureza	Sujeira, lago, sol, aguapés, seres vivos, plantas, microorganismos, natureza, ecossistema, Mãe-Natureza, prefeito, lixeiras, jet-skis, lanchas, ambiente, empresas, resíduos industriais
E	Expressar os conhecimentos sobre um ser vivo	Seres vivos	Peixes, plantas, aguapés, microorganismos
F	Expressar os conhecimentos sobre uma cadeia alimentar	Seres vivos	Cadeia alimentar, peixes, plantas, microorganismos

Quadro 2: Habilidades a serem verificadas.

4. Selecionar os alunos dos grupos A e B, a partir do resultado da avaliação do primeiro questionário: a cada questão, obedecendo ao Quadro 2, foi atribuída uma "nota": (SO) Superou os objetivos; (AO) Atingiu os objetivos; (AP) Atingiu parcialmente os objetivos; (NA) Não atingiu os objetivos; (NR) Não respondeu. Assim, para formar o grupo A, que irá utilizar o jogo e posteriormente realizar nova avaliação, com a finalidade de verificar se o jogo contribui para o ensino de ecologia, foram adotados os seguintes critérios, de acordo com as "notas" obtidas:

4.1. maior quantidade de "Não respondeu – NR";

4.2. seguida da menor quantidade de "Atingiu os objetivos – AO";

4.3. seguida, finalmente, da menor quantidade de "Superou os objetivos – SO".

5. Elaborar novo questionário a ser aplicado no grupo A, após o uso do jogo: o novo questionário foi concebido segundo dois critérios:

5.1. questões do questionário da pré-avaliação (devidamente modificadas) que tiveram a maior quantidade de NR;

5.2. questões de mesma categoria do questionário da pré-avaliação, baseadas no Quadro 2, a fim de completar o número total de questões.

Sendo assim, buscou-se medir habilidades e conceitos trabalhados no jogo construído para esta investigação.

6. Jogar: nesta etapa, o jogo foi disponibilizado aos alunos sob condições controladas de tempo (duas horas). Foram coletados dados sobre a interação e as informações sobre as dificuldades e qualidades apresentadas pelo jogo, por meio da observação e do relato dos alunos durante a atividade.

7. Reaplicar o novo questionário: depois de os alunos do grupo A terem utilizado o *software*, foi realizada nova avaliação.

8.2 Resultados e Análise do Experimento

No questionário de pré-avaliação das crianças do grupo A (Figura 6), 23,82% receberam a "nota" AO; 6,34% foram avaliados com SO; 41,28% não responderam às questões (NR) e 28,55% receberam outras "notas" (AP, NA). A soma das porcentagens de NR com AP e NA demonstra que 69,83% das crianças não mostraram habilidade ou conhecimento suficiente sobre ecologia (de acordo com o definido no Quadro 2).

Entretanto, após o uso do jogo e a aplicação de novo questionário ao mesmo grupo (Figura 7), foi possível verificar alteração positiva em relação à pré-avaliação. A "nota" AO subiu para 36,52%, ou seja, houve um aumento de 12,7%; SO teve um incremento de 3,2%, alcançando 9,5%. Já o resultado NR marcou 14,3%, ou seja, ocorreu uma diminuição de 27% de incidência.

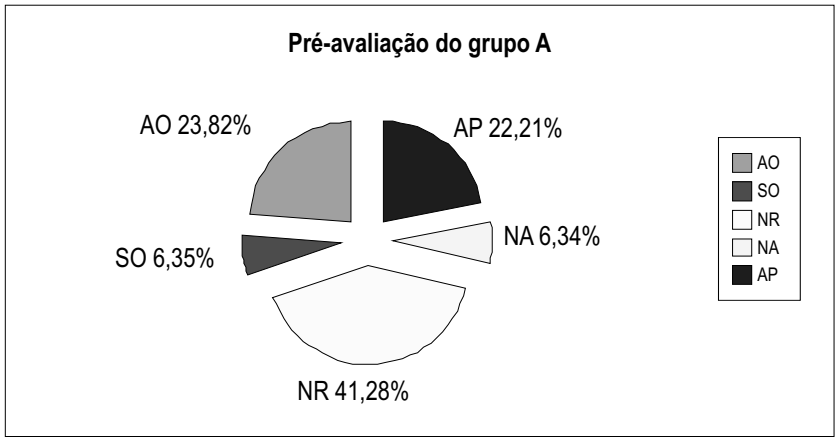

Figura 6: Resultado do primeiro questionário.

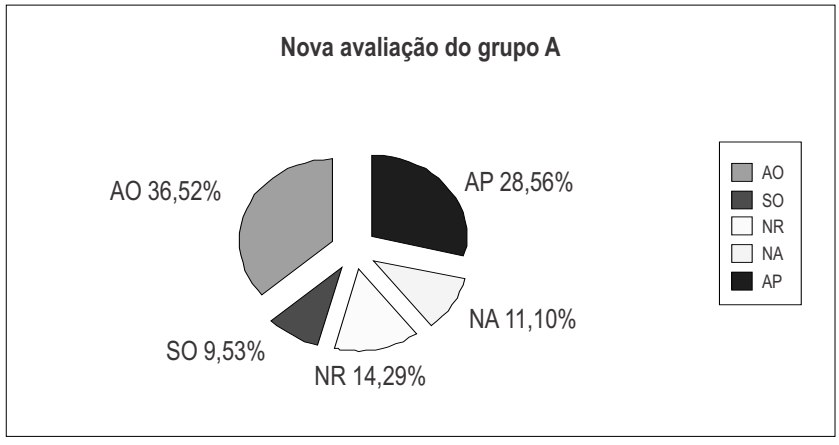

Figura 7: Resultado do segundo questionário.

Considerações Finais

A pesquisa realizada no projeto AS-MCOE aborda técnicas, metodologias e assuntos facilmente adaptáveis e aplicáveis em áreas que necessitam de qualidade nos processos de ensino-aprendizagem e de tomada de decisão. Através da linguagem AgentSpeak(L) e do seu interpretador Jason, é possível especificar comportamentos (em sistemas inteligentes) que, efetivamente, auxiliam em processos de decisão, por meio de planos de ações, a partir da percepção de situações de um específico ambiente, virtual ou real.

No projeto, utilizam-se essas técnicas para modelar os comportamentos de um tutor e de personagens de um jogo. O tutor, a partir de sua especificação, tem condições de auxiliar jogadores enquanto esses interagem com o jogo. Ou seja, conforme ele percebe ou sensora o ambiente, por meio de seus planos, consegue "tomar" decisões (realizar ações) que colaboram com os jogadores, assumindo assim sua função de tutor.

A proposta tecnológica do projeto teve como um dos objetivo reproduzir a modelagem do tutor, especificado inicialmente em X-BDI, em AgentSpeak(L), para que fosse possível a implementação de novas funcionalidades, aplicações e experimentos, que não poderiam ser realizados com a tecnologia original.

O artigo também descreve uma etapa realizada no projeto: a conversão dos agentes implementados em X-BDI para AgentSpeak(L). Fase importante, pois a primeira abordagem apresentava carência de suporte e de manutenção; utilizava *software* proprietário para executar o interpretador; não possibilitava a especificação e a implementação de sistemas multiagentes; não proporcionava o uso em ambientes para a *Web*; além disso, o X-BDI foi atualizado logo após seu uso no trabalho de Giraffa (1999); assim, foi preciso adequar a especificação do tutor na nova sintaxe do X-BDI.

Já nesta nova abordagem, adotada no projeto, ampliam-se as possibilidades: há uma equipe de suporte para o Jason, constantemente realizando correções e atualizações; o interpretador é baseado em *software* livre; por meio do ambiente SACI (*Simple Agent Communication Infrastructure*), o sistema multiagente em Jason pode ser utilizado em redes; é possível especificar e implementar ambientes multiagentes; novamente, pelo interpretador ter sido construído em Java, foi possível sua integração com o *engine* de prototipação de jogos eletrônicos em *software* livre. Enfim, possibilidades que trazem vantagens educacionais: seja no uso de agentes cognitivos (na prática) em STI, construídos na proposta de *software* livre, ou seja na especificação do tutor de forma mais íntegra às teorias educacionais de tutoria artificial.

Este trabalho também apresenta os resultados parciais obtidos com a experimentação de um jogo educacional inteligente para o ensino de ecologia. A proposta original inclui o emprego de um tutor artificial no ensino assistido do conteúdo de ecologia, cuja tecnologia foi modificada com propostas recentes. Contudo, foi necessária uma avaliação à parte do jogo, para que a utilização do tutor possa ser mensurada com maior precisão em um futuro experimento.

Com o experimento relatado, o conhecimento dos alunos pôde ser avaliado de forma quantitativa e qualitativa. Os números apresentados, apesar de não serem representativos (quantidade ideal de sujeitos para um experimento com tais características), demonstraram a evolução dos alunos ao responderem as questões aplicadas. A quantidade de alunos que não responderam (NR) reduziu e houve melhora na qualidade das respostas (de AO para SO). Os motivos para tal conclusão são: (I) maior familiaridade com os termos empregados nos questionários e no jogo; (II) identificação de situações-problema que viabilizaram a contextualização das questões.

Resta, para um novo experimento, avaliar a influência do tutor artificial sobre os alunos que atingiram parcialmente (AP) ou não atingiram os objetivos (NA), e principalmente, sobre os que não souberam responder (NR) às questões apresentadas.

Referências

ASYMETRIX (1994). **ToolBook:** user manual. Bellevue, WA.

Bolzan, W. and Giraffa, L. (2002). Estudo comparativo sobre Sistemas Tutores Inteligentes Multiagentes Web. **Technical Report Series**. Number 024. Faculdade de Informática PUCRS, Porto Alegre.

Bordini, R. H., Vieira, R. (2003). Linguagens de programação orientadas a agentes: uma introdução baseada em Agentspeak(L). **Revista de Informática Teórica e Aplicada**, X(1): 7–38.

Bordini, R. H., Hübner, J. F. (2004). **Jason:** a Java-based AgentSpeak interpreter used with SACI for multi-agen distgribution Over the Net. Disponível em: <http://jason.sourceforge.net/>. Acesso em: junho de 2007.

Bratman, M. (1990). Intention, plans and practical reason. **Harvard University Press**, Cambridge, MA.

Callegari, D. A. (1999). **Aplicando aprendizagem por reforço a uma arquitetura multiagente para suporte ao ensino de educação ambiental**. Dissertação de Mestrado, PPGCC/PUCRS, Porto Alegre.

Cohen, P. and Levesque, H. J. (1987). Intention = choice + commitment. In: **National Conference in AI**, Berlin. Springer-Verlag.

Giraffa, L. M. M. (1999). **Uma Arquitetura de tutor utilizando estados mentais**. Tese de Doutorado, CPGCC/UFRGS, Porto Alegre.

Hübner, J., Bordini, R., Vieira, R. (2004). Introdução ao desenvolvimento de sistemas multiagentes com Jason. In: **XII Escola de Informática da SBC**, volume 2, pág. 51–89, Guarapuava. UNICENTRO.

Móra, M. D. C. (2000). **Um modelo formal e executável de agentes BDI**. Tese de Doutorado, CPGCC/UFRGS, Porto Alegre.

Murray, T. (1999). **Authoring Intelligent Tutoring Systems:** Na analysis of the state of the art. International Journal of Artificial Intelligence in Education.

Raabe, A. L. A.; JAVINCZIK, A. M. ; GIRAFFA, L. M. M. (1996). Eco-Lógico: Ambiente interativo para suporte ao ensino de educação ambiental.. In: **IV Congresso Internacional de Informática e Aprendizagem**.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In: Van Hoe, R., editor, **7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World**, Eindhoven, The Netherlands

Studios, G. T. (2006). **Golden T Game Engine**. Acesso em: Junho de 2006.

Wooldridge, M. (2000). **Reasoning about rational agents**. The MIT Press, London.

Zamberlam, A. O.; Giraffa, L. M. M. (2002). Em direção a uma técnica para programação orientada a agentes BDI. In: **I Workshop de Teses e Dissertações de Inteligência Artificial**, Porto de Galinhas - PE. XVI Brazilian Symposium on Artificial Intelligence and VI Brazilian Symposium on Neural Networks. Recife: SBC/UFPE.